4A

УДК 004.931, 519.174

***А.В. Агарков***
Институт проблем искусственного интеллекта МОН Украины и НАН Украины,
г. Донецк
aav@iai.donetsk.ua

# Поиск множества максимальных клик на основе метода построения дополнительного графа

В работе представлен способ поиска множества максимальных клик в графе. Данный способ основан на методе построения дополнительного графа-пирамиды, что позволяет легко распараллеливать вычисления. Вычислительная сложность представленного способа зависит линейно от количества максимальных клик в графе. При решении конкретных задач (распознавании изображений, например) данный способ позволяет ускорять получение решения. Это достигается за счёт сокращения числа строящихся вершин и рёбер путём использования при их построении дополнительных условий, которые учитывают специфику задачи.

## Введение

В последнее время во многих работах для распознавания изображений используется их описание в виде графов, каждой вершине которого соответствует определённая область изображения [1-5]. При таком подходе поиск объекта на изображении сводится к поиску подграфа, ему соответствующего. Данный подграф выделяется путём сравнения графов описывающих рассматриваемое изображение и искомый объект, что приводит к задаче поиска наибольшей максимальной клики в ассоциированном графе. Однако в более общем случае искомых объектов на изображении может быть более одного, что означает необходимость поиска всех возможных максимальных клик.

Одним из эффективных для поиска множества максимальных клик в графе на данный момент является алгоритм Брона-Кербоша [6]. Его вычислительная сложность линейна относительно количества клик в графе. Однако данный алгоритм не может быть распараллелен, что при современной тенденции к увеличению производительности компьютеров за счёт многоядерности/многопроцессорности и параллельных вычислений является недостатком.

В настоящей работе предлагается способ поиска множества максимальных клик на основе метода построения дополнительного графа [7], [8]. Данный метод сам по себе может быть легко распараллелен, что влечёт за собой возможность распараллеливания и предлагаемого способа. Его вычислительная сложность, так же, как и алгоритма Брона-Кербоша, линейна относительно количества клик в графе.

Далее в работе описаны основы метода построения дополнительного графа-пирамиды, а затем и сам предлагаемый способ, основанный на нём.

**Целью данной работы** является разработка способа поиска множества максимальных клик в графе, который не требует значительных вычислительных затрат, позволяет учитывать специфику решаемых задач и может быть распараллелен.

# Построение дополнительного графа-пирамиды

Пирамидой на графе *G* называется граф *P*, имеющий следующие свойства:

– каждая вершина пирамиды $v_{i,j}$ принадлежит определённому уровню $i$;

– уровни пирамиды упорядочены от нижнего (первого) до верхнего (последнего) ($i = 1,\dots,N$);

– каждой вершине пирамиды $v_{i,j}$ соответствует подграф $p(v_{i,j})$, принадлежащий предыдущему уровню (для вершин начального уровня – это подграфы графа *G*, т.е. предыдущим для первого уровня (т.е. нулевым) является сам граф *G*);

– каждой вершине пирамиды $v_{i,j}$ соответствует подграф $g(v_{i,j})$, принадлежащий графу *G*;

– каждая из вершин подграфа $p(v_{i,j})$ является родителем для вершины $v_{i,j}$, которая, в свою очередь, является дочерней для всех вершин множества $p(v_{i,j})$;

– каждая вершина пирамиды соединена рёбрами (которые называются межуровневыми) со всеми своими родителями;

– $v_{i,j}$ является предком для $v_{l,k}$, а $v_{l,k}$ для $v_{i,j}$ является, соответственно, потомком ($i < l$), если существует цепь, их соединяющая, которая образована только межуровневыми рёбрами, между дочерними и родительскими вершинами, причём не существует вершин, принадлежащих данной цепи, лежащих в общем уровне;

– множество родителей $p(v_{i,j})$ вершины $v_{i,j}$ образуют подграф, который является родительским для вершины $v_{i,j}$;

– вершина пирамиды *P*, не имеющая дочерних вершин, называется верхушкой пирамиды.

Пирамида образуется в результате построения, т.е. начиная с основного графа (нулевого уровня) проводится следующая процедура – граф, соответствующий одному уровню, разбивается на подграфы, для каждого из которых в следующем уровне создаётся новая вершина. Эти новые вершины являются дочерними для вершин соответствующих подграфов предыдущего уровня, которые, в свою очередь, являются родителями для новозаведенных вершин.

На рис. 1 представлен пример пирамиды, построенной на графе, состоящем из вершин a, b, c, d, который представляет собой цепь. Пунктирными линиями обозначены уровни пирамиды. В данной пирамиде три уровня (нулевой уровень – исходный граф). Если рассмотреть отдельную вершину $v_{2,1}$, принадлежащую второму уровню, то для неё родителями являются вершины $v_{1,1}$ и $v_{1,2}$, принадлежащие первому уровню. Дочерней для данной вершины является $v_{3,1}$. Данная вершина имеет с $v_{2,2}$ общего родителя $v_{1,2}$, поэтому $v_{2,1}$ и $v_{2,2}$ соединены ребром. Подграф основного графа, который соответствует $v_{2,1}$, образован вершинами a, b, c.

Вершина $v_{3,1}$ не имеет дочерних вершин, поэтому является верхушкой данной пирамиды, и ей соответствует весь основной граф. Она является потоком для всех остальных вершин этой пирамиды, которые, в свою очередь, являются предками для верхушки.

Таким образом, каждый уровень пирамиды, представляет собой граф, каждой вершине которого соответствует подграф предыдущего уровня. Вершины данного подграфа

соединяются рёбрами, если подграфы, которые им соответствуют, пересекаются. Кроме того, ребрами могут соединяться другие вершины одного уровня, удовлетворяющие правилу построения пирамиды.

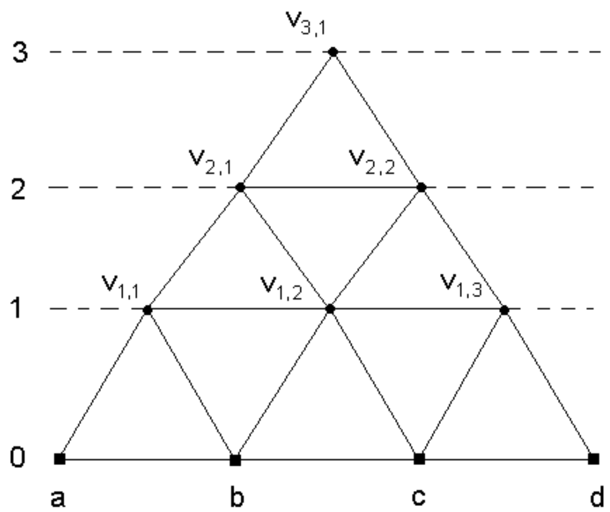



Рисунок 1 – Пример пирамиды

Пирамида может быть построена в соответствии с единым правилом создания вершин нового (следующего) уровня. К примеру, пирамида, представленная на рис. 1, построена по следующему правилу – если две вершины одного уровня соединены ребром, то создаётся вершина следующего уровня, для которой данная пара является родителями; каждая такая пара может иметь только одну дочернюю вершину.

# Построение пирамиды для поиска множества клик

Вид пирамиды и то, какие подграфы соответствуют вершинам пирамиды, зависит от правил её построения. Если необходимо, чтобы данные подграфы обладали определённым свойством, необходимо в качестве подграфов разбиения выбирать такие, которые бы этим свойством обладали. Новые же вершины пирамиды следует соединять рёбрами только в том случае, если объединение соответствующих им подграфов разбиения также обладает необходимыми свойствами.

Так, если целью является поиск множества клик, необходимо, чтобы подграфы разбиения являлись кликами, а новые вершины пирамиды соединялись ребрами только в том случае, если их объединение также является кликой. Если придерживаться данных правил при построении пирамиды, то каждой её верхушке будет соответствовать клика.

Однако этого не достаточно для того, чтобы найденные клики были максимальными и образовывали множество всех клик рассматриваемого графа. В двух случаях будет наблюдаться пропуск решений, т.е. не всем максимальным кликам рассматриваемого графа будут соответствовать вершины пирамиды.

В первом случае это связано с тем, что объединение двух клик может не являться кликой, вследствие того, что некоторые вершины, принадлежащие данным подграфам, могут не быть смежными. Кликой может являться объединение неких подграфов данных клик. Причем, чем больше данные клики – тем больше может быть количество вариантов.

Во втором случае пропуск решений связан с тем, что возможен такой выбор подграфов разбиения, при котором некоторые клики не будут иметь общего потомка, что приведёт к пропуску решений. Для того чтобы не было пропусков решений, необходимо и достаточно, чтобы все клики-тройки оставались в рассмотрении, т.е. имели хотя бы одного общего потомка.

Для того чтобы избежать пропусков решений в первом случае, необходимо учесть все варианты объединения частей подграфов разбиения, которые дают клики. Для этого необходимо подграфы разбиения, в свою очередь, тоже разбивать на более мелкие подграфы и ставить им в соответствие новые вершины пирамиды, что, конечно же, ведёт к росту количества вершин в пирамиде. Поэтому количество вершин в подграфах разбиения должно быть как можно меньше, чтобы сократить количество возможных вариантов.

Наиболее подходящей для решения данной задачи является ребёрная пирамида, у которой в качестве подграфов разбиения выступают пары смежных вершин. То есть каждой вершине данной пирамиды соответствует ребро с инцидентными ей вершинами. Данное обстоятельство позволяет избегать создания новых вершин пирамиды, соответствующих подграфам подграфов разбиения. Действительно, если объединение двух пар смежных вершин пирамиды кликой не является, но является кликой объединение одной пары с только одной вершиной другой пары – это достаточно отразить созданием нового ребра между новой вершиной и единичной вершиной другой пары, что исключит пропуск данной клики. Данная единичная вершина оказывается расположенной как бы на нескольких уровнях пирамиды сразу – это является следствием того, что вместо создания новой вершины пирамиды, соответствующей только одной вершине предыдущего уровня, создается новое ребро, соединяющее вершины разных уровней. Следует отметить, что данные вершины не состоят в родстве (одна не является ни потомком, ни предком другой), хотя и соединены межуровневым ребром. Данному межуровневому ребру соответствует клика из трех вершин одного уровня пирамиды – вершины, инцидентной межуровневому ребру, и пары смежных вершин, инцидентных ребру, на котором построена вершина следующего уровня.

Для того чтобы не было пропусков решений, соответствующих второму случаю, необходимо и достаточно, чтобы все клики-тройки оставались в рассмотрении. Для этого необходимо, чтобы каждой клике-тройке либо соответствовало межуровневое ребро, либо существовало ребро-прообраз для вершины следующего уровня, объединение с которым данной клики-тройки являлось также кликой.

Если построение новых вершин пирамиды начинать с создания межуровневых рёбер, то это позволяет унифицировать алгоритмы для создания рёбер между вершинами пирамиды и избежать пропусков решений первого случая.

Действительно, рассмотрим построение ребёрной пирамиды для поиска множества максимальных клик, с использованием межуровневых рёбер. Для этого обозначим:

$P = (V^P, E^P)$ – пирамида;

$E_0$ – множество рёбер пирамиды, которые могут быть использованы для построения новых вершин;

$E_1$ – множество рёбер пирамиды, которые использовались для построения новых вершин пирамиды;

$E_2$ – множество рёбер пирамиды, которые входят в подграфы-клики, соответствующие объединению клик, вершины которых соединены ребром;

$V_0$ – множество вершин пирамиды, инцидентных рёбрам из множества $E_0$;

$p(v_i) = (v_j, v_k), p_1(v_i) = v_j, p_2(v_i) = v_k$ – пара родительских вершин вершины $v_i$;

$e(v_i) = e(v_j, v_k)$ – ребро, на котором построена вершина пирамиды $v_i$;

$g(v_i)$ – подграф исходного графа $G$, который соответствует вершине пирамиды $v_i$;

$T$ – множество всех клик-троек пирамиды;

$Q$ – множество всех клик пирамиды;

$E_I$ – множество межуровневых рёбер пирамиды;

$E_i + = e_j$ – операция добавления ребра $e_j$ в множество $E_i$;

$E_i - = e_j$ – операция удаления ребра $e_j$ из множества $E_i$;

$a = A + +$ – создание нового элемента $a$ множества $A$.

Условие образования межуровневого ребра между вершинами пирамиды $v_i$ и $v_j$, где $v_i$ – вершина более высокого уровня, чем $v_j$, имеет вид:

$$\left(p(v_i) \cup v_j\right) \in T \Rightarrow e(v_i, v_j) = E^P + +. \quad (1)$$

Условие образования ребра между вершинами одного уровня $v_i$ и $v_j$ имеет вид:

$$\left(p(v_i) \cup p(v_j)\right) \in Q \Rightarrow e(v_i, v_j) = E^P + +, \quad (2)$$

что при условии наличия уже образованных межуровневых рёбер можно переписать в виде

$$\left[\left(v_i \cup p(v_j)\right) \in T \vee \left(p(v_i) \cup v_j\right) \in T\right] \Rightarrow e(v_i, v_j) = E^P + +.$$

Причём очевидно, что

$$\left(v_i \cup p(v_j)\right) \in T \Leftrightarrow \left(p(v_i) \cup v_j\right) \in T,$$

поэтому условие (2) можно переписать в виде:

$$\left(p(v_i) \cup v_j\right) \in T \Rightarrow e(v_i, v_j) = E^P + +,$$

что совпадает с (1). Таким образом, для построения новых рёбер достаточно использовать одно условие, что даёт возможность применения только одного алгоритма. Необходимо, однако, вначале соединять новые вершины со старыми, а потом с друг другом, т.е. начинать построение новых рёбер с межуровневых.

При образовании нового ребра $e(v_i, v_j)$ оно автоматически добавляется в множество $E_0$, а ребра $e(v_i, p_1(v_j))$, $e(v_i, p_2(v_j))$, $e(v_j, p_1(v_i))$, $e(v_j, p_2(v_i))$ удаляются из $E_0$ и добавляются в $E_2$. Это препятствует образованию новых вершин $v_i$ и $v_j$, для которых $g(v_i) \subseteq g(v_j) \wedge g(v_j) \subseteq g(v_j)$, что позволяет избегать наличия вершин, соответствующие графы которых совпадают.

Если после построения всех новых рёбер остались межуровневые рёбра, принадлежащие множеству $E_0$, то это означает, что объединение подграфов, отвечающих двум соответствующим новым вершинам, кликой не является. Однако является кликой объединение их частей, т.е. учитывается первый случай, описанный выше, и не происходит пропуска решений.

Таким образом, построение межуровневых рёбер позволяет упростить процесс построения рёбер между новыми вершинами пирамиды и избежать попуска решений, связанных с первым случаем.

Для того чтобы избежать пропусков решений, связанных со вторым случаем, необходимо выбирать множество $E_1$ таким образом, чтобы выполнялось условие для клик-троек:

$$\forall t \in T\, \exists e = e(v_i, v_j) \in E_1 : \left(t \cup (v_i, v_j; e)\right) \in Q. \quad (3)$$

Следует также заметить, что рёбра для множества $E_1$ должны выбираться таким образом, чтобы в каждой клике-тройке было не более одного такого ребра

$$\forall t \in T \neg \exists e_i, e_j \in E_1 : e_i \neq e_j ; e_i, e_j \in t . \tag{4}$$

Это условие необходимо для того, чтобы избежать образования вершин пирамиды с совпадающими соответствующими подграфами, т.е. для того, чтобы избежать повторных решений.

Обозначим

$T(e(v_i, v_j)) = \{t_k \in T : (v_i, v_j) \in t_k\}$ – множество клик-троек, которым принадлежит ребро $e$ и пара инцидентных ей вершин.

Условие добавления ребра $e$ в множество $E_1$:

$$e \in E_0, \neg \exists t \in T(e) : \exists e_i \in E_1, e_i \in t \Rightarrow E_0 - = e, E_1 + = e . \tag{5}$$

То есть при соблюдении данного условия ребро удаляется из множества $E_0$ и добавляется в множество $E_1$. Перебрав таким образом все ребра из множества $E_0$ формируется множество $E_1$. Для того чтобы обозначить принадлежность рёбер к тому или иному множеству, каждому их них присваивается соответствующий маркер. Таким образом, перенос ребра из одного множества в другое сводится к смене маркера данного ребра.

При таком способе заполнения $E_1$ условие (4) будет выполнено в силу правила формирования данного множества.

Однако выполнение условия (3) сужает множество возможных вариантов для $E_1$, что усложняет его формирование. Для того чтобы обойти эту трудность, используется следующий приём – если для рассматриваемого ребра $e$ (5) не соблюдается, но соблюдается условие

$$e \in E_0, \exists t_i \in T(e) : \neg \exists t_j \in T(e) : \exists e_l \in t_j, e_l \in E_1 ; (t_i \cup t_j) \in Q , \tag{6}$$

то создаётся ребро $e_N = e(v_i, v_j) : e = e(v_i, v_j)$, которое добавляется в множество $E_1$. То есть создаётся копия рассматриваемого ребра, которая и добавляется в $E_1$. А само ребро $e$ из множества $E_0$ переносится в множество $E_2$ – $E_0 - = e, E_2 + = e$. При создании новых рёбер, инцидентных новым вершинам пирамиды, учитывается смежность, соответствующая только рёбрам из множеств $E_0$ и $E_2$. Этот приём позволяет оставить в рассмотрении все клики-тройки, не допустить создания вершин с совпадающими соответствующими подграфами и не создавать специального алгоритма выбора рёбер для множества $E_1$.

Таким образом, учитывая всё вышесказанное, построение одного уровня пирамиды выглядит следующим образом:

1) формирование множества $E_1$ из рёбер, принадлежащих множеству $E_0$;

2) создание новых вершин на основе рёбер из множества $E_1$;

3) создание новых рёбер, инцидентных новым вершинам с одновременным формированием множества $E_0$.

Построение пирамиды ведётся до тех пор, пока множество $E_0$ остаётся непустым.

Построение пирамиды $P(V^P, E^P)$ начинается с её инициализации исходным графом $G(V, E)$, т.е. перед началом создания первого уровня пирамиды выполняется отношение $P(V^P, E^P) = G(V, E)$. Также перед началом создания первого уровня все рёбра пирамиды добавляются в множество $E_0$, т.е. в начале $E_0 = E^P$.

Обозначим

$T(e) = \{t_i : t_i \in T, e \in t_i\}$,

$V_T\big(e(v_i, v_j)\big) = \{v_k : v_k \in E^P, \{v_i, v_j, v_k\} \in t \in T(e)\}$,

$e_i^j$ – $i$-е ребро принадлежащее множеству $E_j$,

$N(A)$ – мощность множества $A$,

$E_1(e) = \{e_i : e_i \in E_1, e \in t \in T(e_i), e_i \neq e\}$.

С учётом введенных обозначений условие (5) переписывается в виде:

$$E_1(e_i^0) = \varnothing,$$

а условие (6) – в виде:

$$\exists t_i \in T(e) : \neg\exists t_j \in T(e_l \in E_1(e)), (t_i \cup t_j) \in Q.$$

Тогда алгоритм формирования множества $E_1$ из рёбер, принадлежащих множеству $E_0$, на псевдокоде записывается в следующем виде:

*GetE*$_1$

{

  $for(i = 1,..., N(E_0))$ // цикл по всем $e_i^0 \in E_0$

  {

    $if\left(\exists t_l \in T(e_i^0) : \neg\exists t_j \in T(e_k \in E_1(e_i^0)), (t_l \cup t_j) \in Q\right)$

    {

      $if\left(E_1(e_i^0) = \varnothing\right)$

      {

        $E_0 -= e_i^0$;

        $E_1 += e_i^0$;

      }

      *else*

      {

        $e^N = E^P ++$;

        $e^N = e(v_n, v_m : e_i^0 = e(v_n, v_m))$;

        $E_0 -= e_i^0$;

        $E_2 += e_i^0$;

        $E_1 += e^N$;

        $T(e^N) = T(e_i^0) / (T(e_i^0) \cap (\bigcup_{e_k \in E_1(e_i^0)} T(e_k)))$;

        $V_T(e^N) = V_T(e_i^0) / (V_T(e_i^0) \cap (\bigcup_{e_k \in E_1(e_i^0)} V_T(e_k)))$;

      }

    }

  }

}

Алгоритм создания новых вершин на основе рёбер из множества $E_1$:

*GetNewVertexs*

{

  $V_N^P = \varnothing$; // множество новых вершин пирамиды;

$for(i = 1,...,N(E_1))$
$\{$
$v^N = V^P ++;$
$p(v^N) = (v_j, v_k : e_i^1 = e(v_j, v_k));$
$V_N^P += v^N;$
$\}$
$\}$

Алгоритм создания новых рёбер, инцидентных новым вершинам с одновременным формированием множества $E_0$:

$GetNewEdgesAndE_0$
$\{$
$for(i = 1,...,N(V_N^P))$ // цикл по всем $v_i^N \in V_N^P$
$\{$
$e_p = e(v_i^n);$ // $e \in E_1$
$e_p = e(v_j, v_k);$
$for(i = 1,...,N(V_T(e_p)))$ // цикл по всем $v_l^T \in V_T(e_p)$
$\{$
$if(\neg\exists e(v_i^N, v_l^T))$
$\{$
$e^N = E^P ++;$
$e^N = e(v_i^N, v_l^T);$
$E_0 += e^N;$
$\}$
$e^1 = e(v_j, v_i^N);$
$e^2 = e(v_k, v_i^N);$
$E_0 -= e^1;$
$E_2 += e^1;$
$E_0 -= e^2;$
$E_2 += e^2;$
$if\left(\exists e(v_l^T, p_1(v_i^N)), e(v_l^T, p_2(v_i^N))\right)$
$\{$
$e^3 = e(v_l^T, p_1(v_i^N));$
$e^4 = e(v_l^T, p_2(v_i^N));$
$E_0 -= e^3;$
$E_2 += e^3;$
$E_0 -= e^4;$
$E_2 += e^4;$
$\}$
$\}$
$\}$
$\}$

Таким образом, алгоритм построения пирамиды для поиска множества максимальных клик имеет вид

$BuildPyramid$

$\{$

$P(V^P, E^P) = G(V, E)$;

$E_0 = E^P$;

$while\,(E_0 \neq \varnothing)$

$\{$

$GetE_1$;

$GetNewVertexs$;

$GetNewEdgesAndE_0$;

$\}$

$\}$.

После окончания построения пирамиды её верхушкам, т.е. вершинам, которые являются смежными только со своими родителями, будут соответствовать максимальные клики. Количество верхушек в пирамиде равно количеству всех максимальных клик исследуемого графа $G(V, E)$. Количество предков отдельной верхушки есть $O(N_q)$, где $N_q$ – число вершин в клике, соответствующей рассматриваемой верхушке.

При формировании множества $E_0$ рассматриваются все рёбра пирамиды, количество которых оценивается как $O(N_q{}^2)$. Для того чтобы провести проверку одного ребра, необходимо проверить все клики-тройки, которым оно принадлежит. Количество таких клик для одного ребра составляет $O(N_q)$. Таким образом, вычислительная сложность формирования множества $E_1$ и создания вершин пирамиды, относящихся к одной максимальной клике, составляет $O(N_q{}^3)$.

При создании рёбер, инцидентных одной вершине, необходимо проверить все смежные вершины родителей, количество которых оценивается как $O(N_q)$. То есть вычислительная сложность создания вершин пирамиды, относящихся к одной максимальной клике, составляет $O(N_q{}^2)$. Вычислительная сложность создания подграфа пирамиды, соответствующей одной максимальной клике, составляет $O(N_q{}^3) + O(N_q{}^2) = O(N_q{}^3)$.

Если количество всех максимальных клик составляет $N(Q_{\max})$, а среднее количество вершин в них оценить как $N(E)/N(V)$, где $N(E)$ и $N(V)$ – количество соответственно рёбер и вершин в исследуемом графе, то оценка для вычислительной сложности построения пирамиды составит $O\left(N(Q_{\max})\left(N(E)/N(V)\right)^3\right)$.

Таким образом, вычислительная сложность построения пирамиды для поиска множества максимальных клик зависит линейно от их количества.

## Выводы

Предложенный в данной работе способ на основе метода построения дополнительного графа-пирамиды позволяет эффективно проводить поиск множества максимальных клик. Его вычислительная сложность, так же, как и алгоритма Брона-Кербоша, зависит линейно относительно количества максимальных клик. Однако в отличие от

данного алгоритма предложенный способ легко распараллеливается. Действительно, создание новых вершин и рёбер может осуществятся независимо друг от друга, т.е. параллельно.

Следует также заметить, что при решении задач поиска и распознавания изображений (объектов) данный способ позволяет использовать дополнительные условия, отражающие особенности искомого решения, при создании новых вершин и рёбер пирамиды. Это позволяет сократить время распознавания, что достаточно важно для построения систем реального времени и поиска по большим базам данных.

Таким образом, предложенный способ поиска множества максимальных клик в графе, обладает рядом свойств, делающих его полезным для решения задач распознавания образов, представленных в виде графов.

# Литература

1. Alireza Ahmadyfard. Region-Based Object Recognition: Pruning Multiple Representations and Hypotheses / Alireza Ahmadyfard, Josef Kittler // British Machine Vision Conference. – 2000. – P. 745-754.
2. Karin Kailing. Content-Based Image Retrieval Using Multiple Representations / Karin Kailing, Hans-Peter Kriegel, Stefan Schonauer // Proc. 8th Int. Conf. on Knowledge-Based Intelligent and Engineering System. – Wellington, New Zealand. – 2004. – LNAI 3214. – P.982-988.
3. Feng Tang. Object tracking with dynamic feature graph / Feng Tang, Hai Tao // 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. – 2005. – P. 25-32.
4. Агарков А.В. Структурное описание изображений в виде графа для решения задач распознавания / А.В. Агарков // Бионика интеллекта. – 2009. – № 1(70). – С. 95-101.
5. Агарков А.В. Сегментация изображения на основе его описания в виде графа / А.В. Агарков // Искусственный интеллект. – 2010. – № 3. – С. 274-282.
6. Bron C. Algorithm 457 – Finding all cliques of an undirected graph / C. Bron, J. Kerbosh // Comm. of ACM. – 1973. – № 16. – P. 575.
7. Агарков А.В. Метод сравнения двух графов за полиномиальное время / А.В. Агарков // Искусственный интеллект. – 2003. – № 4. – С. 172-184.
8. Агарков А.В. Поиск изоморфных пересечений двух графов за полиномиальное время / А.В. Агарков // Искусственный интеллект. – 2007. – № 2. – С. 62-74.

# Literatura

1. Ahmadyfard A. British Machine Vision Conference. 2000. P. 745-754.
2. Kailing K. Proc. 8th Int. Conf. on Knowledge-Based Intelligent and Engineering System. Wellington, New Zealand. 2004. LNAI 3214. P. 982-988.
3. Feng Tang. 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. 2005. P. 25-32.
4. Agarkov A.V. Bionika intellekta. №.1 (70). 2009. S. 95-101
5. Agarkov A.V. Iskusstvennyj intellekt. № 3. 2010 S. 274-282.
6. Bron C. Finding all cliques of an undirected graph, Comm. of ACM. 16. P. 575.
7. Agarkov A.V. Iskusstvennyj intellekt. № 4. 2003. S. 172-184.
8. Agarkov A.V. Iskusstvennyj intellekt. № 2. 2007. S. 62-74.

***A.V. Agarkov***

**Search of the Set of Maximal Cliques Based on the Method for Constructing Complementary Graph**

This paper presents a method of finding the set of maximal cliques in a graph. This method is based on the method for constructing complementary graph-pyramid that makes it easy to parallelize computations. The computational complexity of the given method linearly depends on number ofmaximal cliques in a graph. For specific tasks (for example, image recognition) this method stimulates solutions. This is achieved by reducing number of constructed vertices and edges by the additional conditions in their construction, which take into account characteristics of the tasks.

*Статья поступила в редакцию 14.06.2011.*